2017

Real-Time Fall Detection and Response on Mobile Phones Using Machine Learning

Ilona Shparii

Loyola University Chicago

LOYOLA UNIVERSITY CHICAGO

REAL-TIME FALL DETECTION AND RESPONSE ON MOBILE PHONES USING MACHINE LEARNING

A THESIS SUBMITTED TO

THE FACULTY OF THE GRADUATE SCHOOL

IN CANDIDACY FOR THE DEGREE OF

MASTER OF SCIENCE

PROGRAM IN COMPUTER SCIENCE

BY

ILONA SHPARII

CHICAGO, IL

AUGUST 2017

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to my academic advisor and mentor Dr. Mark V. Albert for giving me such an amazing opportunity to work on this research project. Regardless of his busy schedule he always had time to answer any questions or discuss details of the project. He was the first person who introduced me to the world of machine learning and guided me through my research and academic journey. His valuable advice helped me in all of the aspects of this study and beyond it and I cannot imagine having a better advisor.

I am grateful to Dr. Dmitriy Dligach and Dr. Earvin Balderama for agreeing to be my committee members and for challenging my work. I would like to thank Loyola University Chicago and the Rehabilitation Institute of Chicago (formerly, RIC, but now known as the Shirley Ryan Abilitylab as of March 2017) for providing funding for my research work.

Especially, I thank Dr. Arun Jayaraman and his Rehabilitation Technologies and Outcomes (RT&O) lab at the Rehabilitation Institute of Chicago for allowing me to be a part of this truly incredible research project that started as a summer 2016 internship. Many special thanks to the team I was working with at RIC: Dr. Luca Lonini, Nicholas Shawen, and Chaithanya K Mummidisetty. As this project is highly collaborative, it could not have been done by one person. Their thoughtful discussions and rich experience taught me a lot and supported my thesis efforts.

We closely worked with The Center for Behavioral Intervention Technologies (CBITs) at Northwestern University team and Chris Karr, former Head of Mobile Research & Development at CBITs, who developed Purple Robot application. I truly appreciate their valuable input that helped us to achieve our goals.

Last, but not least, I cannot express enough how invaluable my family is to me. They supported me all the way through my educational journey until this moment. I would not be at this point without them.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

LIST OF TABLES vi

LIST OF FIGURES vii

ABSTRACT viii

CHAPTER I: INTRODUCTION 1
Motivation 1
Fall Detection Strategies 3
Wearable Sensors Used for Fall Detection 3
Fall Detection Algorithms 5
Threshold Based Algorithms 5
Machine Learning for Fall Detection 6
Previous Work 8

CHAPTER II: METHODOLOGY 13
Experimental Design 13
Data Preprocessing 17
Feature Extraction 18
Fall Detection Algorithm 20
Model Comparison 20
Favored Model 21
Real-Time Fall Detection System 24

CHAPTER III: RESULTS 28
Model Comparison 28
Subject Population Models 30
Device Location Models 32
Comparison with a Threshold-Based Algorithm 33
Evaluation Using Real-World Data 35

CHAPTER IV: DISCUSSION 36
Relation to Previous Work 36
Limitations 39
Real-Time Fall Detection Challenges 40
Future Efforts 42
Conclusion 43

REFERENCE LIST 45

VITA 49

LIST OF TABLES

Table 1. Features extracted from each accelerometer and gyroscope readings 19

Table 2. Features extracted from barometer readings 20

Table 3. Hyperparameters considered for different classifiers 22

Table 4. Optimal hyperparameter values for different classifiers 29

Table 5. Classifier performance results 29

Table 6. Summary results of subject population models 30

Table 7. Confusion matrices for subject population models 31

Table 8. Summary results of device location model performance with the machine learning approach 32

Table 9. Confusion matrices for device location models with the machine learning approach 33

Table 10. Summary results of device location models performance with threshold-based algorithm 34

Table 11. Confusion matrices for device location models with the threshold-based algorithm 34

Table 12. Summary results of device location model performance on continuously recorded data outside the lab environment 35

LIST OF FIGURES

Figure 1. Phases of fall analyzed by threshold-based algorithms 5

Figure 2. The scope of NIH R01 “Understanding Real-Life Falls in Amputees using Mobile Phone Technology” project conducted by Max Nader Lab for Rehabilitation Technologies and Outcomes Research of Rehabilitation Institute of Chicago and supervised by Arun Jayaraman 12

Figure 3. Healthy control subject performs simulated falls with a phone placed in pouch on a waist: a) trip b) slip 14

Figure 4. A screenshot of the Purple Robot mobile application labeling tool 16

Figure 5. The workflow of estimating performance of a fall detection model 24

Figure 6. The workflow of the real-time fall detection system implemented as a part of the Purple Robot mobile application 25

Figure 7. The screenshots of Purple Robot application showing the FallNetProbe 26

Figure 8. The screenshots of the Purple Robot application showing a FallNetProbeFailure 27

Figure 9. The screenshots of the Purple Robot application showing a triggered Fall Detection 27

Figure 10. Accelerometer readings of amputee falls that were misclassified as non-falls 31

Figure 11. Accelerometer readings of amputee activities that were misclassified as falls 31

ABSTRACT

Falls are common and often dangerous for groups with impaired mobility, like the elderly or people with lower limb amputations. Finding ways of minimizing the frequency or impact of a fall can improve quality of life dramatically. When someone does fall, real-time detection of the fall and a long-lie can trigger fast medical assistance. Such a system can also collect reliable data on the nature of real-world falls that can be used to better understand the circumstances, to aid in prevention efforts. This work has been to develop a real-time fall tracking system specifically for subjects with lower limb amputations.

In this study 17 subjects (10 healthy controls and 7 amputees) were asked to simulate 4 types of falls (trip, slip, right and left lateral) 3 times each with a mobile phone placed at 3 different locations on the body (pouch, pocket, and hand). Signals were collected from the accelerometer, gyroscope and barometer sensors using the Android mobile phone application Purple Robot. We compared 5 different machine learning classifiers for fall detection: logistic regression (L^1 and L^2 norm), support vector machines, K-nearest neighbors, decision trees, and random forest. Logistic regression (L^1 regularized “lasso”) and random forest yielded the best results on the test set (98.8% and 98.4%, respectively). There was no significant difference between amputee and healthy control falls in terms of classifier accuracy. When testing on real world data with no recorded falls, the false positive rate was only 0.07%.

In addition to offline algorithmic development, the detection system was implemented for real-time application on a mobile platform. The previously-trained logistic regression model was implemented on the mobile platform for real-time detection. This platform will be used in an upcoming amputee population falls study. The completed system will gather data on the current conditions leading to the fall (weather, GPS location, etc.) and classify the type of the fall. The system will follow up with notifications requesting a response from the user, or automatically notify emergency contacts or 911 as needed. The steps taken in creating this system bring us closer to real-time intervention strategies to minimize the impact of falls, and enable us to collect accurate falls-related data to improve fall prevention strategies and prosthesis design.

CHAPTER I

INTRODUCTION

Motivation

Falls are a very common occurrence in the elderly and especially among those with lower limb amputations; around 30% of people after age 75 fall at least once per year (Sixsmith and Johnson, 2004). Falls are also the primary cause of injury-related death in the elderly, amounting to 55% of all injury-related deaths (Fulks et al., 2002; Kramarow et al., 2015). Falls are more frequent in subjects with lower-limb amputations; in one study 52.4% of amputees fell at least once over the past year; 75% of these individuals fell twice or more (Miller et al., 2001).

There are nearly 2 million people living with limb loss in the United States, and 185,000 amputations occur each year. The leading cause of limb amputation is dysvascular disease such as diabetes and peripheral arterial disease (Amputee Coalition, 2017). Due to the rise of obesity and related illnesses, limb loss due to dysvascular disease is on the rise, with the number of lower-limb amputees expected to double in approximately 40 years (Ziegler-Graham et al., 2008). Due to the frequency of falls in this growing population, understanding and minimizing the impact of falls is a critical societal health concern.

Aside from physical injuries that falls might cause (such as fractures, non-healing wounds, and bleeding disorders) psychological factors should also be taken into the

account. In MacBride et al. 58% of subjects reported a fear of falling - the main concern among all the participants (McBride et al., 1980). In addition, fear of falling might lead to a decrease of independence, decline of mobility, lowering of one's self-esteem and/or discouraging use of prosthesis (Ryynanen et al.,1992; Spice et al., 2009; Vellas et al., 1997; Mann et al., 2006; Delbaere et al., 2004).

It is well known that falls and fear of falling might have serious consequences on one's health and overall well-being. Common risk factors of falls are balance instability, functional impairments, chronic disease, mental status, and use of medications (Miller et al., 2001). The amputee population is at high risk of falling as one or more of these factors might be true for them. Nevertheless, the main focus of falls research has primarily been the elderly population (Sixsmith and Johnson, 2004; JS, 2002; McBride et al., 1980; Ryynanen et al.,1992; Spice et al., 2009; Vellas et al., 1997; Mann et al., 2006; Delbaere et al., 2004). To date, our understanding of falls in amputees is very limited and we strongly believe that more research needs to be conducted.

Most information about falls in amputees that exists today is based on retrospective data or questionnaires which were filled some time after a fall had occurred. Obviously, such information relies on a patient's memory, which might be imprecise or incomplete. Detecting falls automatically will allow us to collect more reliable data. It will give an opportunity to switch from a descriptive analysis of falls to a more quantitative, analytical approach that can be used for development of fall-prevention techniques.

Kulkarni et al. described three main categories of risk factors for falls: prosthesis, personal, and environmental factors. It is believed that each category might be

connected to a different type of fall (trip, slip or lateral) or cause a different type of injury (Kulkarni et al., 1996). We believe that having a real time fall detection system which will collect reliable data will help to better understand the reasons and circumstances which lead to a fall. Better understanding can influence prosthetic design and current guidelines for fall-prevention strategies.

Fall Detection Strategies

There are many ways to detect falls. From a broad perspective fall detection systems could be classified as environmentally smart systems and wearable devices. Environmental systems use externally deployed sensors such as cameras, floor sensors, infrared sensors, microphones and/or pressure sensors. Wearable devices use sensors placed on a subject's body, such as accelerometers or gyroscopes, on devices ranging from fitness wearables to mobile phones. We will focus on the wearable devices approach. In order to do fall detection those systems involve two main components: hardware (sensors) and software (algorithm) which we will not describe.

Wearable Sensors Used for Fall Detection.

Many fall detection studies use dedicated wearable sensors, primarily triaxial accelerometers (Bagala et al., 2012; Shi et al., 2012; Shany et al., 2012; Zhang et al., 2011; Ying et al., 2011; Tolkiehn et al., 2011). Using wearable sensors is a convenient way to obtain reliable motion data related to falls, in contrast to the subjective recall of self-reports. Most wearable sensors are lightweight and small, some of them are only millimeters in size; they are highly portable and relatively inexpensive. Accelerometers provide information about subject movement through accelerations of the device during

those movements. In addition, they can show relative orientation to the direction of gravity, which can be extremely useful for fall detection.

The gyroscope is another valuable sensor for fall detection. Whereas accelerometers measure linear acceleration, gyroscopes measure angular acceleration. Given the complementary nature of these two types of motion, these two types of sensors are often used in conjunction. Together, they have the potential to boost accuracy for tasks such as fall detection.

Current wearable devices are able to measure the number of floors climbed in an elevator as well as estimate elevation changes primarily from the use of barometers - sensors used to measure air pressure. In fact, barometers present in a current smart phone (iPhone 7) can reliably detect changes of up to 1 meter in height due to the slight variation in air pressure with small increases in elevation, providing a useful, though less reliable signal with smaller changes in height. Because falls commonly end with a change in the height of the device, this can be a useful signal for indication of a fall.

To date, modern smartphones have accelerometers, gyroscopes, barometers, GPS and many more sensors built-in. This makes a phone a really convenient platform for activity recognition or fall detection. They are widely used and affordable for most people. Almost everyone is familiar with how to use a smartphone today. Compared to separate wearable sensors, smartphones have better data storage capabilities and more powerful batteries. Thus, smartphones have advantages that make them more suitable for real-time fall detection that can be used outside of a laboratory setting.

Fall Detection Algorithms.

There are two main approaches to analyzing data from wearables sensors: threshold-based algorithms and machine learning techniques. We will describe the key points of both below.

Threshold-based Algorithms. Fall events can be divided into 3 phases (figure 1): pre-fall (activity done before a fall), impact (the moment of falling), and post-fall (activity after a fall). A fall can be detected by observing if one or more features of the sensor readings exceeds a predefined threshold. The most common features for these threshold-based algorithms are fall acceleration, fall velocity, fall impact, and orientation after the fall (Bagala et al., 2012).

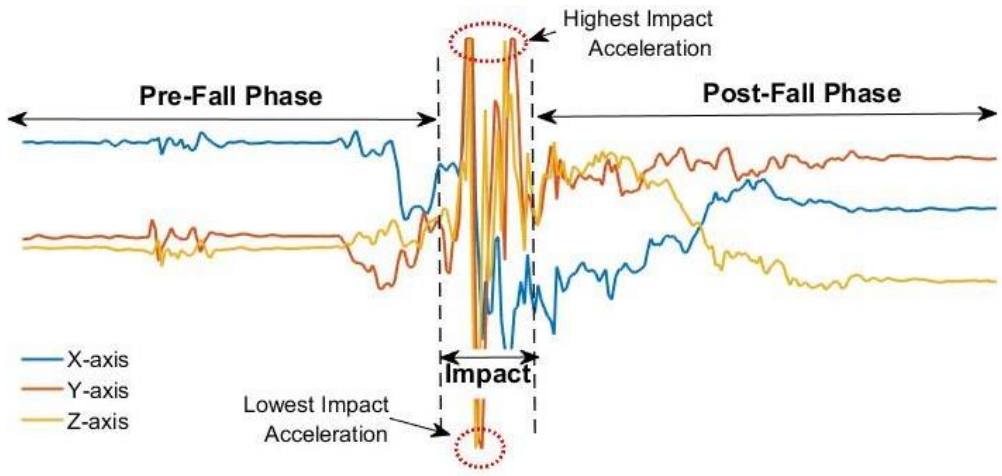

Figure 1. Phases of fall analyzed by threshold-based algorithms. *The plot displays accelerometer readings of a side fall performed by a healthy volunteer with a phone placed in a pouch (waist).*

Bagala et al. compared performance of different threshold-based algorithms and found that most of them are tuned using simulated falls data and are therefore not suitable for real-world application (Bagala et al., 2012). Importantly, algorithms that had

better fall detection rates on the simulated falls tended to have higher false alarm rates when applied to real-world data. Errors of this type can be expected as these threshold-based algorithms are simply classifying events with a maximum magnitude as falls, while in practice there are many more events that can meet such thresholds due to high acceleration. Jumping and “falling” into a bed can easily be mistaken as falls by most fall detection systems. Threshold-based algorithms are especially prone to these kinds of errors.

Machine Learning for Fall Detection. Use of machine learning techniques can significantly improve fall detection as it is a more sophisticated approach compared to threshold-based algorithms. Instead of specifying particular features and applying a simple threshold, training data is provided to a model so that the model itself can determine the best weighted combination of features to perform the classification. While it may sound relatively simple to do, there are certain challenges when using machine learning such as determining which features to use, choosing the predictive model, finding the right parameters for that model, and objectively validating that model’s performance.

It is important to choose the right features for a successful classification model. Generally speaking, features are readily computable characteristics of a given sample of data which can possibly improve prediction. They do not need to be highly predictive individually, but a series of weak features when combined can sometimes be quite powerful for classification. The features can be directly measured data or derived measures from a sample (calculated mean, standard deviation, extreme values, etc.).

Besides manually picking features based on intuitions of what could be helpful there are a few automated methods that can aid the feature selection process. For example, some classifiers, like logistic regression, naturally rate feature utility by assigning non-zero coefficients to useful features and zero-valued coefficients to useless features. For models that don't have a direct means for measuring the utility of features, there are alternate strategies available to measure the importance of features - e.g. measuring the improvement in prediction accuracy with/without a given feature. In any case, feature selection is an important process and can be done by intuition or automatically using machine learning classifiers that can handle high-dimensional feature sets well.

Having too many or too few features can lead to classic modeling concerns including overfitting and overgeneralization. Overfitting occurs when an overly complex model is used, for example, fitting a straight line of points with a high-order polynomial. Having a model that is too complex for the given problem can lead to models that fit to noise in a given data set, rather than the relevant structure for the problem at hand. Overgeneralization, on the other hand, is when a model is too simple model - e.g. using a line to fit points along a curve. Both overfitting and overgeneralization are common concerns with fitting prediction models and either extreme would cause poor model performance on new data sets.

Different models can be more suitable for certain types of problems than for others, that's why model choice is also important. For example, some models are known to perform poorly with a relatively large number of features. Others make certain statistical assumptions about the problem - like naive bayes assuming the features are statistically independent. Support vector machines, logistic regression, naive bayes,

k-nearest neighbors, decision trees, and multi-layer neural networks have been previously used for fall detection (Albert et al., 2012; Abbate et al., 2012). Given the ubiquity of machine learning classification libraries, it can be relatively straightforward to apply multiple models to a given data set, and select the appropriate one based on performance.

In order to get the best model performance, available data is usually split into at least two sets: one for training and the other for testing. The testing set should not be involved in the training process in any way to avoid overfitting. Instead of picking arbitrary divisions for training and test data, cross-validation can be used to automate the process. The most popular ways to perform cross-validation are K-fold, leave-one-out, and subject-wise cross-validation. With K-fold cross-validation the training dataset is split into K parts; then the model is trained on K-1 parts and tested on the remaining part that was separated. The process is repeated K times with a different test set at each iteration. Leave-one-out cross validation is the same as K-fold where K is equal to number of data samples. When cross-validation is done for each subject as a separate fold, it is subject-wise cross-validation. Subject-wise cross-validation is used to estimate the behavior of a trained model on a novel subject. By using cross-validation appropriately, we are able to select the better features, models, and model hyperparameters to be applied on the untouched test data.

Previous Work

Our study takes advantage of previous work on fall detection and classification and extends the traditional approach from offline data analysis to building a fully

functional real-time Android based fall detection system, specifically designed for the amputee population.

Albert et al. used falls data collected in a lab setting from 15 subjects for fall detection and classification (Albert et al., 2012). They collected data with a smartphone worn on a belt always in the same position such that accelerometer x, y, and z-axis corresponded to the up, left, and backward directions of the subject. Having a consistent orientation of the sensor makes different types of falls more distinguishable. For example, if someone trips and falls forward, the z-axis would point the same direction as gravity.

In order to be classified, the falls data had to be preprocessed first. The raw accelerometer signal was split into 2 second clips. Then, an extended set of signal processing features known to work well in activity recognition were extracted from each clip. Five different machine learning classifiers were applied to this data set to detect the type of fall that occurred among slips, trips, and lateral falls: support vector machines (SVM), logistic regression (lasso), naïve bayes, k-nearest neighbors, and decision trees. They were able to achieve almost perfect classification among a set of falls types – 99% with SVM and logistic regression classifiers using 10-fold cross-validation which dropped only by 0.5% for SVM when tested using subject-wise cross-validation.

In addition, they attempted to do fall detection to distinguish fall events from fall-like events. Fall-like events were defined as daily activities with the highest acceleration changes taken from a set of 1 week recordings. Such events could be easily misclassified as falls, especially in threshold-based algorithms. Training with such a data set is likely to improve performance on real-world data compared to the use of

other low-impact activities of daily living used as non-fall examples. SVM and logistic regression had the best results out of all the classifiers - 98% with 10-fold cross-validation and 97% when subject-wise cross-validation was used.

In the research presented in this thesis, we will extend the analysis and develop a fall detection system designed for real-time use by people with lower-limb amputations. A new data set was collected from both healthy controls and amputees. To the best of our knowledge, there are no studies that used amputee data for fall detection.

In most studies the wearable device is always located in the same place and orientation on the body, which is an unrealistic expectation for real-world mobile phone use. In this study we introduced a variety of locations a phone could be used: pocket, pouch, and hand. Additionally, we did not instruct participants about the orientation of the phone during the data collection process. Clearly, this creates a more challenging fall detection task, but having data that is varied in the same ways that real-world collected data can be is more likely to lead to a model that robust enough to handle real-world scenarios.

We take the feature set that has been proven to work for both activity recognition and fall detection/classification in previous studies (Albert et al., 2012; Albert and Kording, 2011; Albert et al., 2013) and build an improved machine learning fall detection model capable of working in real-time for the amputee population. The Android application that has been built will be used to collect valuable real-world amputee falls data - setting that stage for further research focused on fall prevention.

This thesis work is only part of the team effort for the full NIH R01-funded project "Understanding Real-Life Falls in Amputees using Mobile Phone Technology" conducted by Max Nader Lab for Rehabilitation Technologies and Outcomes Research of the Rehabilitation Institute of Chicago (as of March 2017, now the Shirley Ryan Abilitylab), which is supervised by Arun Jayaraman (Jayaraman, 2014).

The main idea is to develop a real-time fall detection system on the Android mobile platform. Such a system would be designed specifically for people with lower-limb amputations. As modern smartphones have built-in motion sensors including accelerometers, gyroscopes, and barometers, their signals can be successfully used to detect a fall by applying machine learning techniques. The system will perform the following functions (Figure 2): while a subject wears a phone on the body (waist or pocket) or holds it in hand, the sensor data is evaluated every five seconds. If a fall occurs, the system will alert the user, immediately send the fall information and related data in a brief window of time to a server (including sensor data 15 minutes before and after the fall). The server will continuously process received data, classify the type of a fall, gather environmental information such as weather, GPS location, and activity that was done prior to the fall to generate a report. If there is a high likelihood of a fall being detected, we will ask a user to confirm on the device. If they respond positively, the generated report will be sent to their physician. If they don't respond, the system is capable of contacting an identified emergency contact, or if neither of them respond, dial 911. If a user indicates that a fall did not occur after a fall event is triggered, the system will report a 'false positive' alert which can be used to continuously improve

future models. The scope of this thesis work is the real-time fall detection system on the Android device. All server-side processing is considered future efforts of the R01 team.

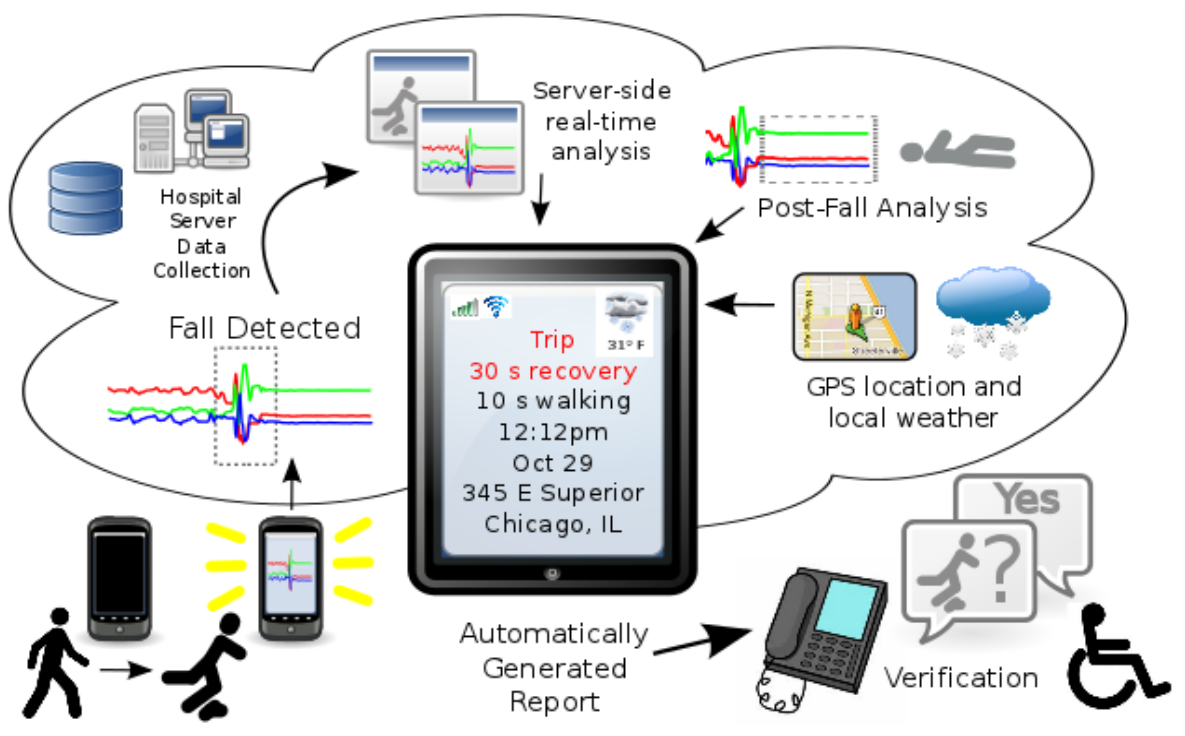

Figure 2. The scope of NIH R01 “Understanding Real-Life Falls in Amputees using Mobile Phone Technology” project conducted by Max Nader Lab for Rehabilitation Technologies and Outcomes Research of Rehabilitation Institute of Chicago. *Adapted from Jayaraman, A. (2014). Understanding Real-Life Falls in Amputees using Mobile Phone Technology. Project # 1R01EB019406-01A1.*

CHAPTER II

METHODOLOGY

The methods that have been used in this study will be described in the same sequence as they have been applied: experimental design, data preprocessing, feature extraction, and model selection. Additionally, we describe the platform used for the implementation of the real-time fall detection system.

Experimental Design

The overall research design was initially developed by the Max Nader Rehabilitation Technologies and Outcomes Research Center of the Rehabilitation Institute of Chicago (RT&O lab). The overarching design goals were part of a successfully funded National Institutes of Health (NIH) R01 grant “Understanding Real-Life Falls in Amputees Using Mobile Phone Technology” - project number 1R01EB019406-01A1.

Two groups of subjects were recruited: people with lower-limb amputations (7 individuals) and healthy controls (10 individuals). All the subjects were screened with appropriate inclusion and exclusion criteria and consented prior to participating in the study. The criteria included that a subject must be between 18 and 85 years inclusive, willing to carry and use a smartphone as their primary communication device, be willing to give written consent, and able to comply with study procedures. Additional expectations for the amputee population are that a subject has a unilateral amputation

of the lower limb above or below the knee within at least six months after first fitting of a lower extremity prosthesis, and they must use either a mechanical or a microprocessor-controlled prosthesis in activities of daily living.

All participants were asked to simulate 4 types of falls (trips, slips, right and left lateral) 3 times each with a phone placed at 3 different locations (pouch, pocket and hand) - 36 falls per subject. Subjects were instructed to fall as closely to real-world scenarios as possible and use any protective strategies they would normally use. For example, a person walked towards a mat and then tripped over it falling forward (Figure 3a). Similarly, they fell backwards onto the mat simulating a slip (Figure 3b). For right and left lateral falls subjects were pushed sideways onto a mat. Although the phone was placed in one of three locations on the body generally, there was some variability in each general location. For example, the pouch was always worn on the waist, but could be placed at any side of the body, or even directly in front. Similarly, participants were free to choose a pants pocket (side or back) based on their preference.

Figure 3. A healthy control subject performs simulated falls with a phone placed in a pouch on the waist performing a trip (left) and slip (right).

To ensure safety of the participants they were asked to wear specially designed protective gear which was first tested on healthy volunteers. The gear is a combination of different pieces of protective clothing that are worn by sports players: football shoulder pads, neck rolls, back plates and rib protection; skateboard elbow pads, knee pads, wrist guards, and bumsaver shorts; soccer shin guards; hockey helmet, pants and jersey. The maximum coverage is achieved in this setup by safeguarding all the body parts that are most vulnerable during a fall including head, neck, elbows, knees, and wrists.

Subjects were also asked to perform a set of activities of daily living (ADLs) such as sitting, standing, walking, lying, and stair climbing (up and down) with the same variability in the phone placement location (pouch, pocket, and hand). While holding a phone in the hand subjects were instructed to use it, for instance, browsing the internet or checking messages. In later analysis, all the ADL data, including transitions between them, was combined and labeled as non-falls for the purpose of fall detection.

In addition, in order to collect data of more natural daily movements, 3 amputees were asked to carry a phone in a pocket for at least 2 days. No participant fell during this period, so all collected data represented non-falls. This data was used to evaluate the fall detection model’s false positive rate in real-world scenarios.

The data was collected using Samsung Galaxy S4 phones running Android version 4.4.4 with the Purple Robot mobile application developed by the Center for Behavioral Intervention Technologies (CBITs) at Northwestern University. Purple Robot provides a convenient way for collecting information about the user and their immediate surrounding using real-time sensors data (Purple Robot, 2017).

When collecting data two phones were used simultaneously: one for recording sensory signals on the subject and the other for labeling the data by the experimenter. The first phone was placed on the subject's body recording accelerometer, gyroscope, and barometer signals at a variable sampling rate, approximately 50 Hz on average. The data from this phone was automatically split into 5 second clips, interpolated with a cubic polynomial to exactly 50 Hz, and transmitted to the server as soon as the phone was able to access the web.

While a subject was performing falls and ADLs a researcher was labeling data using the Purple Robot labeling tool on the other phone. The labeling tool was specifically designed for our project allowing us to mark start and end time points of each activity and attach other properties to the label such as subject ID, type of the fall, location where phone is placed, etc. (Figure 4).

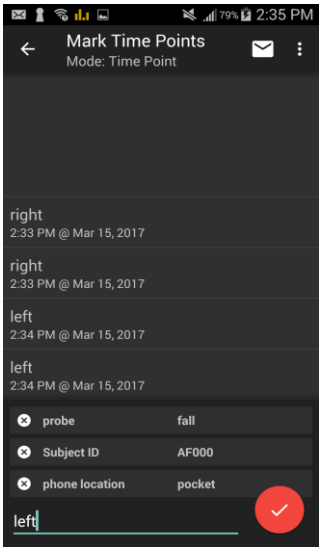

Figure 4. A screenshot of the Purple Robot mobile application labeling tool. *The red button triggers stopwatch-like functionality recording start and end time. Record shown on the screen are created labels. Panels on the bottom of the screen are properties with additional information that are attached to a label.*

The information with labels was also transmitted to the server and could be downloaded later for the analysis.

In total, we collected 283 instances of amputee falls along with 2268 clips of amputee ADLs. Similarly, 335 instances of healthy control falls were collected along with 2272 clips of healthy control ADLs.

Data Preprocessing

The data from both subject and experimenter phones (containing sensors readings and labels) are downloaded from the server as one file in text format. Each record in the file contains a phone id, probe name (kind of data: fall label, activity label or data related to a 5 second clip), timestamp when the event was logged and a payload in JSON format associated with it. A payload contains accelerometer (timestamp, X, Y, Z), gyroscope (timestamp, X, Y, Z), barometer (timestamp, altitude, pressure) readings, and other statistical information (e.g. minimum and maximum timestamps associated with sensors events) related to the 5 seconds that were evaluated (see ‘Real-Time Fall Detection System’ section for detailed information on the evaluation process).

To make data processing faster we wrote a script that organizes data from a single data collection session into a MATLAB format for analysis. Additionally, the script visualizes all fall events and activities for the given session providing the ability for manual data quality control.

For each fall, we formed a 10 second clip centered at the beginning of the fall. To represent the fact that the fall may occur anywhere within a clip, we resampled each 10 second fall clip with a sliding 5 second window so the fall appears at different locations (beginning, middle, end). The resampling was done based on a uniform random

distribution such that the start of the fall can occur anywhere in the interval from 0 to 3 seconds with a uniform probability. In this way, one 10 second clip would produce ten 5 second clips from a single fall event with the fall present at different locations in each clip. Data clips that contain ADLs were kept in the original format without any shifting or resampling.

As a result, the data preprocessing stage resulted in data ready for analysis containing sensors readings and labels associated with 5 seconds clips for falls (shifted and resampled 10 times) and ADLs.

Feature Extraction

Our feature set was based on a set previously proven to work in activity recognition and fall detection/classification problems (Albert et al., 2012; Albert and Kording, 2011; Albert et al., 2013). We also added other values that, in our opinion, might contribute useful information for distinguishing fall events from non-falls. This led to 5061 total features per clip (2477 extracted from the accelerometer, 2477 from the gyroscope, and 107 extracted from barometer readings). In order to minimize the amount of on-board signal processing, initial models were created to assess groups of features. Using an early collected subset of the data, an elastic net model (a logistic regression variant explained later) was used to assess the strength of features. Following this, the feature set was reduced to 1211 features per clip (602 accelerometer, 602 gyroscope, and 7 barometer as shown in Tables 1 and Table 2). We note the potential for overfitting by performing feature selection on the same set of data used for testing, but given the selected features represent 24% of the original set, the

concern for overfitting is less likely and outweighed by the practical concern of having a feature set which could be calculated on the phone in real-time.

Feature	Description	Number of Features
Moments	mean, median, standard deviation, skew, kurtosis: x,y,z	15
Correlation Coefficients		6
Fast Fourier Transform of entire signal	first half (lower frequencies) of 50 bins: x,y,z	75
Fast Fourier Transform of each second	5 bins per 5 seconds: x,y,z	75
Derivatives Moments	mean, median, standard deviation, skew, kurtosis: x,y,z	15
Fast Fourier Transform of Derivatives of entire signal	first half (lower frequencies) of 50 bins: x,y,z	75
Fast Fourier Transform of Derivatives of each second	5 bins per 5 seconds: x,y,z	75
Moments of the Resultant Vector	mean, median, standard deviation, skew, kurtosis	5
Extremes	max of absolute values: x,y,z; min and max of the resultant vector and its derivatives	7
Interquartile Range	x,y,z, the resultant vector and its derivative	5
Extremes Range	x,y,z	3
Angle Moments	mean, IQR, standard deviation, skew, kurtosis of four-quadrant inverse tangent: xy, xz, yz	15
Angle Extremes	max, min of four-quadrant inverse tangent: xy, xz, yz	6
Entropy	x,y,z, derivatives, FFT of each axis and FFT of derivatives	12
Cross Product Statistics	mean of absolute values, standard deviation, skew, kurtosis, median, IQR, max, min	24

Derivatives Cross Product Statistics	mean of absolute values, standard deviation, skew, kurtosis, median, IQR, max, min	24
FFT Energy	mean, standard deviation, skew, kurtosis: x,y,z on 5 bins	60
FFT Entropy	x,y,z on 5 bins	15
Derivatives FFT Energy	mean, standard deviation, skew, kurtosis: x,y,z on 5 bins	60
Derivatives FFT Entropy	x,y,z on 5 bins	15
Derivatives Signal Energy	mean, standard deviation, skew, kurtosis, entropy of the resultant vector	15
Total		**602**

Table 1. Features extracted from each accelerometer and gyroscope readings.

Feature	Number of Features
Standard Deviation: pressure	1
Correlation Coefficient: pressure	1
Fast Fourier Transform of each second: pressure	5
Total	**7**

Table 2. Features extracted from barometer readings.

All features were computed on each 5 second clip creating a set of a standard “samples x features” matrix for training by available machine learning classifiers.

Fall Detection Algorithm

In this section we will briefly discuss the testing of different machine learning models, followed by a more thorough description of the favored model.

Model Comparison.

Because the data was preprocessed into a standard form for submission to machine learning classifiers, we were able to apply a number of classifiers to compare performance. The following are classifiers were considered:

- K-nearest neighbors

- Support vector machines (SVM)
- Logistic regression with ridge penalty
- Logistic regression with lasso penalty
- Decision tree
- Random forest

First we separated data into a training set and a testing set. For the test set, the data from 4 subjects was used (2 amputees and 2 healthy); For the training set, the data from the remaining 13 subjects was used. The training set was used for feature selection and hyperparameter tuning, while the test set was only used for comparison between models.

To simplify comparison we performed additional feature selection - this was done using importance estimates from the extra-trees classifier. Features that influenced the model by less than 10% were disregarded. This led to only 128 features from each sample, from the previously selected 1211 features. Then, for hyperparameter selection, we used grid search technique with a leave one subject out (LOSO) cross-validation approach. Table 3 shows the range of hyperparameter values that were considered for different models.

Favored Model.

When different models were compared, random forest and logistic regression with the lasso regularization penalty (L^1-norm) had the best performance (see results section). Since the ultimate goal is to have a real-time smartphone-based fall detection system, we chose logistic regression as the implementation model as it would be simpler to evaluate and deploy in a mobile environment; while random forest would require

implementing many decision trees, logistic regression requires updating only a set of coefficients. Because elastic net is a generalized linear model which incorporates both lasso (L^1 norm) and ridge regression (L^2 norm) at the extremes (with alpha = 0 and alpha = 1, respectively), it is a superior model to lasso or ridge separately (Zou and Hastie, 2005). For this reason, the superior performance of lasso regression led to our selection of the elastic net classifier in the real-time implementation.

Classifier	Hyper-parameter	Hyperparameter Range
K-nearest neighbors	n neighbors	[1, 3, 5, 7, 9, 11, 13, 15]
SVM	C - penalty parameter	[1e-3, 1e-2, …, 1e+9, 1e+10]
Ridge logistic regression	C - penalty parameter	[1e-3, 1e-2, …, 1e+9, 1e+10]
Lasso logistic regression	C - penalty parameter	[0.001, 0.01, 0.1, 1, 10, 100]
Decision tree	min samples split	[2, 4, 6, 8, 10]
	min samples leaf	[1, 5, 10, 15, 20]
	max depth	[10, 20, 30, 40, 50]
Random forest	n estimators	[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200]

Table 3. Hyperparameters considered for different classifiers.

The elastic net model uses a set of coefficients corresponding to each feature in order to calculate the probability that a given set of features is a fall event. The coefficients in elastic net regression are picked by minimizing the following loss function:

$$min_{\beta}\left(\frac{1}{2N}\sum_{i=1}^{N}(y_i-\boldsymbol{\beta}^{T}\boldsymbol{x}_i)^2+\lambda(\alpha\sum_{p=1}^{P}|\beta_p|+(1-\alpha)\sum_{p=1}^{P}\beta_p{}^2)\right)$$

where β is a vector of model coefficients; N - number of samples; x - input (feature) vector; y - response vector (y_i = 0 non-falls, 1 falls); $\lambda \geq 0$ - regularization parameter (larger values make regularization stronger); $0 \leq \alpha \leq 1$ - coefficient that sets compromise between the ridge and lasso penalty with 0 being ridge regression and 1 being lasso.

We fit 10 values of λ (the smallest λ=4.125e-05 and the largest λ=0.4125) and 10 values of α (0.1, 0.2, …, 1) using 10 fold cross-validation on all available data. The models were fairly robust to changes in the hyperparameters λ and α. The values used for the presented results were λ = 0.015 and α = 0.6.

The model performance was estimated by using either LOSO cross-validation or an external testing set (Figure 5). The quantitative performance indicators we chose are precision, recall and overall accuracy. Precision shows how many identified samples are correct (fraction of true positives over sum of true positives and false positives) while recall indicates how many true samples from a class are correctly identified (fraction of true positives over sum of true positives and false negatives). Overall accuracy can be defined as a description of systematic errors and is calculated as the sum of true positives and true negatives divided by the total number of all samples.

Most of the analysis steps described above were done offline using various desktop development and analytics tools including Matlab 2016b (MathWorks), Python 3.5 using the Anaconda 1.6 package (Continuum Analytics), and Java 8 (Oracle).

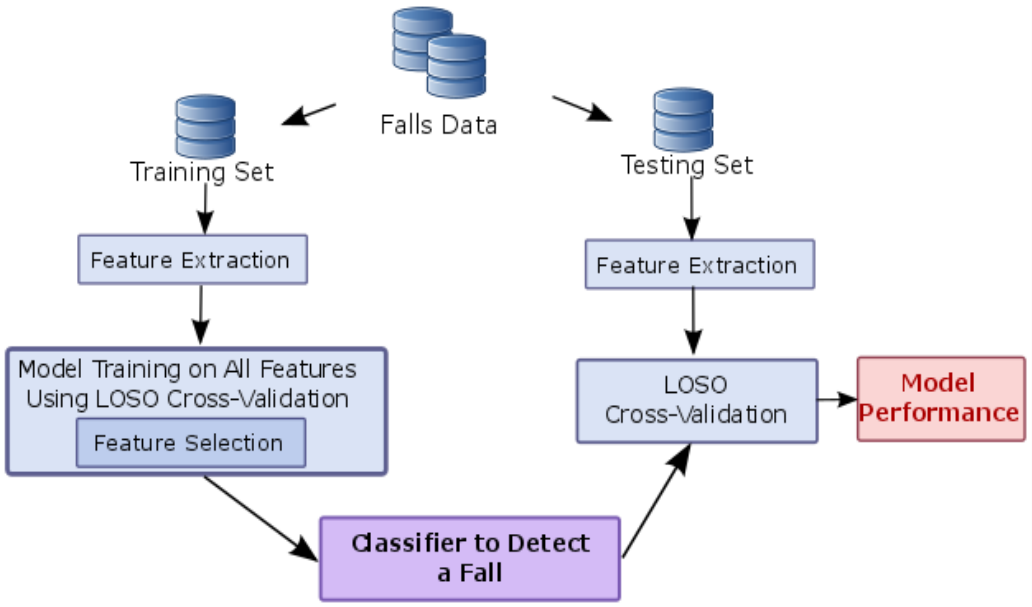

Figure 5. The workflow of estimating performance of a fall detection model.

Real-Time Fall Detection System

Our real-time fall detection system was implemented as a part of the Purple Robot application that was mentioned in the 'Experimental Design' section. As we closely worked with the CBITs research group, they developed customizations for the Purple Robot app to address our needs using an incremental design approach. This allowed us to collect and preprocess data more efficiently as some of the steps were implemented on a phone (See 'Experimental Design' and 'Data Preprocessing' sections).

The real-time recognition system works in the following way (Figure 6): the accelerometer, gyroscope, and barometer signals are processed every 5 seconds. The raw accelerometer and gyroscope data is interpolated to 50Hz with a cubic polynomial. Then, the clip must pass the data quality criteria to be considered valid (the number of samples per clip should be not less than 200 for accelerometer and gyroscope, not less

than 10 for barometer, and the gap between each 2 consecutive samples in accelerometer and barometer readings should be no more than 200 milliseconds). If the data is valid, the evaluation process begins. The same set of 1211 features that was used during off-line analysis is extracted from the clip. The model was already trained offline and saved as a MATLAB file with model coefficients. The application uses the model coefficient file and applies the model to features extracted from the clip. The model yields the probability that the clip is a fall, which is compared against preset threshold (we used a threshold equal to 0.5) to determine whether a fall occurred.

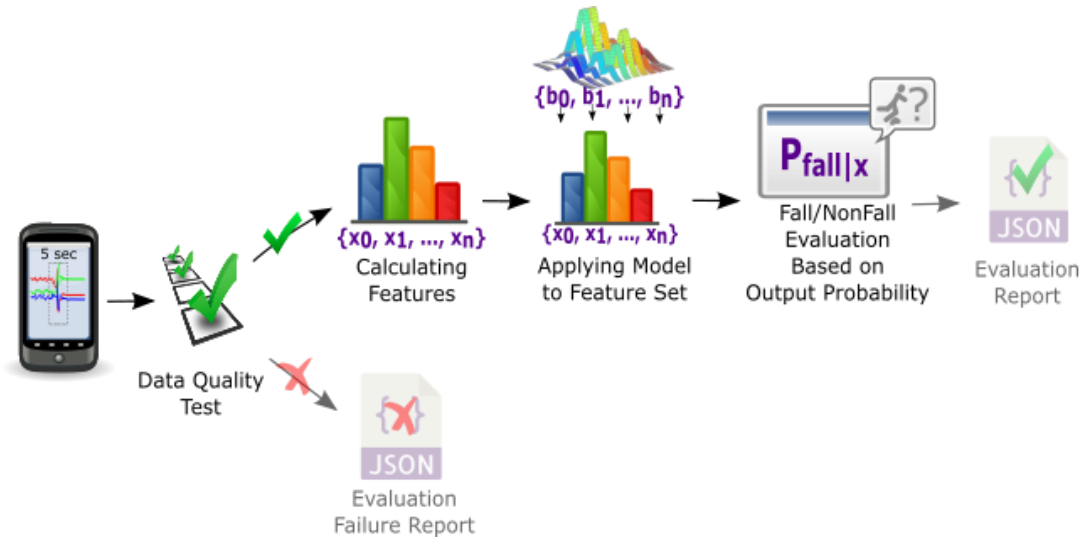

Figure 6. The workflow of the real-time fall detection system implemented as a part of the Purple Robot mobile application. *The accelerometer, gyroscope and barometer signals are split into 5 second clips. Each clip must pass data quality criteria in order to be evaluated. If the clip passes, features are calculated on the sensors signals. Then, the model is applied yielding a probability that the clip represents a fall. The evaluation information is then sent to a server in a JSON format as a FallNet probe. If the clip fails, the FallNetFailure report is sent to the server.*

As the Purple Robot evaluates sensor recordings every 5 seconds, the information about valid clips (Figure 7) or clips that were not valid (Figure 8) appears on the main screen of application along with other probes triggered by the application that

are part of Purple Robot functionality, but are not related to fall detection. A user can click on any probe name to see the properties of the probe. For example, the successful evaluation of the data window (FallNetProbe) has attached information about whether it corresponds to a fall or non-fall, fall probability, fall coefficients, raw sensors values, timestamps, times spent on different evaluation stages, etc. A failure to evaluate the clip (FallNetProbeFailure) contains information about the failure reason, quantitative measurement of the value that failed data quality testing, and timestamps related to different evaluation stages. If fall model detects a fall, it triggers an alert with a prompt asking the user to confirm if fall occurred (Figure 9). As we are still dealing with a relatively high false positive rate the functionality related to the user's response has not been implemented yet. The user interface will also be improved as the deployment stage approaches.

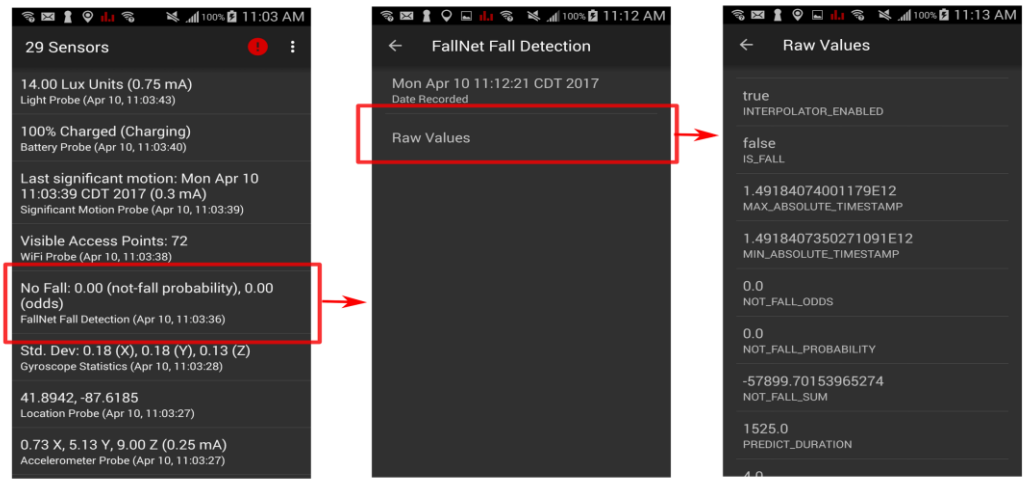

Figure 7. The screenshots of Purple Robot application showing the FallNetProbe. *The first screenshot from the left is the main screen of the application showing FallNetProbe (highlighted) along with other probes that are part of Purple Robot but not the fall detection model. When one clicks on the probe, the information expands to the whole screen (middle screenshot). FallNetProbe has the date-time and raw value properties. When 'Raw Values' is clicked, the detailed information is opened on the whole screen (right screenshot).*

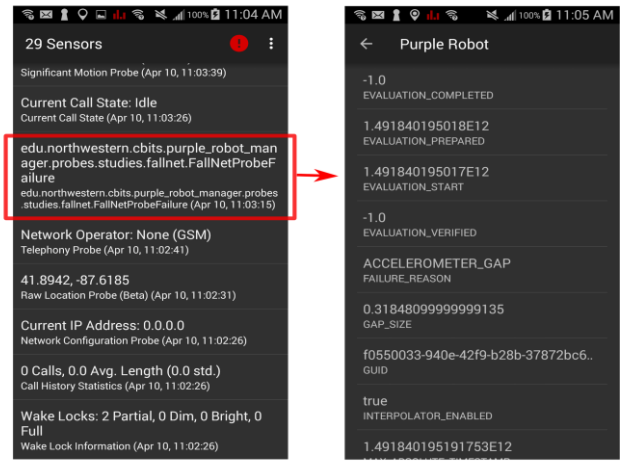

Figure 8. The screenshots of the Purple Robot application showing a FallNetProbeFailure. *The first screenshot from the left is a main screen of the application showing a FallNetProbeFailure (highlighted) along with other probes that are parts of Purple Robot but not the fall detection model. When a probe is clicked, it expands to the whole screen (right screenshot) showing its properties.*

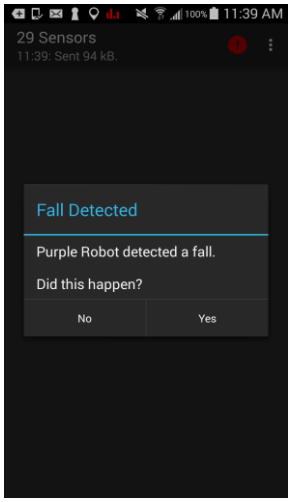

Figure 9. Screenshot of the Purple Robot application showing a triggered Fall Detection. *When a fall is detected the confirmation prompt appears on the screen.*

CHAPTER III

RESULTS

This chapter contains the fall detection results by stages of analysis: model comparison, subject population models, device location models. Comparison was also made with a simple threshold-based algorithm, and the resulting model was also evaluated using real-world data.

Model Comparison

As it was described in the methods section, we have tried 6 machine learning classifiers on the same data set. The hyperparameters were picked based on the results of a grid-search using the leave-one-subject-out (LOSO) cross-validation approach (Table 4).

Logistic regression with the lasso penalty and random forest yielded the best results - 98.77% and 98.44% overall accuracy, respectively. Table 5 shows precision, recall, and overall accuracy for all tested classifiers. Due to the superior performance of lasso regression, the model selected for all later sections in this chapter was elastic net regression - a generalized linear model which is equivalent to lasso regression when α = 1. Additional explanation and reasoning behind the selection of elastic net regression is available in the methods section.

Classifier	Hyper-parameter	Optimal Value
K-nearest neighbors	n neighbors	9
SVM	C - penalty parameter	10 000 000
Ridge logistic regression	C - penalty parameter	10 000
Lasso logistic regression	C - penalty parameter	1
Decision tree	min samples split	2
	min samples leaf	1
	max depth	10
Random forest	n estimators	20

Table 4. Optimal hyperparameter values for different classifiers.

Model	Precision		Recall		Overall Accuracy
	Fall	Non-Fall	Fall	Non-Fall	
K-nearest neighbors	96.57%	98.16%	99.81%	98.17%	96.72% (95.55%-97.64%)
SVM	99.37%	73.57%	94.51%	96.26%	94.75% 92.95%-96.21%)
Ridge logistic regression	96.70%	98.82.0%	98.85%	78.51%	96.93% (95.46%-98.02%)
Lasso logistic regression	99.44%	93.92%	99.16%	95.86%	98.77% (97.98%-99.31%)
Decision tree	98.57%	77.84%	96.55%	89.65%	95.73% (94.44%-96.80%)
Random forest	99.07%	93.75%	99.16%	93.10%	98.44% (97.57%-99.06%)

Table 5. Classifier performance results. *The ranges of overall accuracy were computed as symmetric 95% confidence intervals from the Binomial distribution.*

Subject Population Models

We explored the effect of training data to testing data by subject population in each (healthy controls and amputees) on the classifier performance. Three models were compared: Healthy (training) to Healthy (testing), Healthy to Amputee, and Amputee to Amputee. The results provided for Healthy to Healthy and Amputee to Amputee models are based on LOSO cross-validation which wasn't necessary on the Healthy to Amputee validation (Table 6). The data from all 3 phone locations were combined. The overall accuracy of Healthy to Amputee (98.60%-99.19%) was not significantly lower than from Healthy to Healthy (99.11%-99.55%) or Amputee to Amputee (98.71%-99.27%) models. Therefore, we can conclude that the data from healthy controls could be used for training to detect a fall in the amputee population.

Model	Precision		Recall		Overall Accuracy
	Fall	Non-Fall	Fall	Non-Fall	
Healthy-Healthy	99.08%	99.78%	99.85%	98.64%	99.36% (99.11%-99.55%)
Healthy-Amputee	98.80%	99.29%	99.26%	98.50%	98.92% (98.60%-99.19%)
Amputee-Amputee	98.53%	99.61%	99.68%	98.23%	99.02% (98.71%-99.27%)

Table 6. Summary results of subject population models. *The ranges of overall accuracy were computed as symmetric 95% confidence intervals from the Binomial distribution.*

Confusion matrices for all population models are provided in Table 7. Out of all falls that were not detected by the model, trips and slips were the most common - 58%

and 39% respectively. Figure 10 and Figure 11 display examples of accelerometer readings of falls and non-falls that were misclassified.

Healthy-Healthy			Healthy-Amputee			Amputee-Amputee		
Predicted / True	Non-falls	Falls	Predicted / True	Non-falls	Falls	Predicted / True	Non-falls	Falls
Non-falls	2241	31	**Non-falls**	2234	34	**Non-falls**	2326	42
Falls	5	3349	**Falls**	21	2811	**Falls**	9	2823

Table 7. Confusion matrices for subject population models.

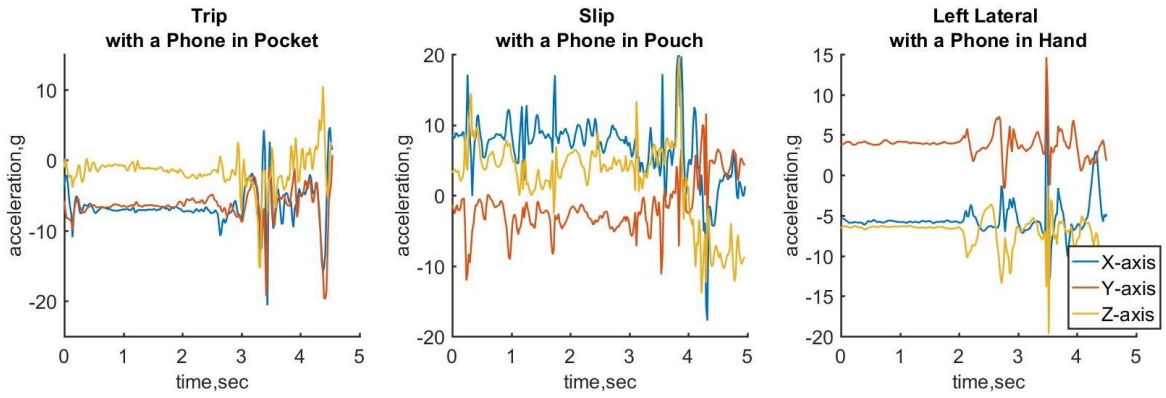

Figure 10. Accelerometer readings of amputee falls that were misclassified as non-falls.

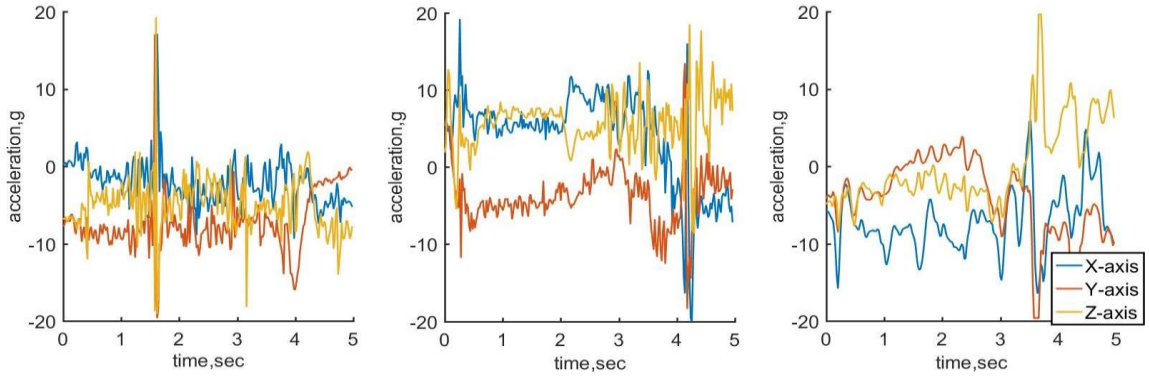

Figure 11. Accelerometer readings of amputee activities that were misclassified as falls.

Device Location Models

To estimate the effect of device location on model performance we trained the model on data from different locations and tested on data from all locations. In this experiment data from healthy controls was used for training and amputee data for testing, as both populations were previously shown to fall in similar ways. The highest accuracy was achieved when the model was trained on data from all locations (98.50%-99.22%) while the lowest was trained using hand-only data (95.35%-96.68%) but not significantly different from the models trained only from the pocket or pouch data (Table 8 and Table 9). An interesting note is that although the location-specific models all have similar overall accuracies, the pouch and pocket models have higher false positive rates while the hand model has a higher false negative rate by comparison.

Model	Precision		Recall		Overall Accuracy
	Fall	Non-Fall	Fall	Non-Fall	
Pouch (Waist)	94.60%	100.0%	100.0%	93.32%	96.92% (96.29%-97.47%)
Pocket	94.50%	99.93%	99.95%	93.19%	96.83% (96.20%-97.39%)
Hand	98.87%	93.11%	93.75%	98.75%	96.06% (95.35%-96.68%)
All Locations	98.99%	98.81%	98.99%	98.81%	98.91% (98.50%-99.22%)

Table 8. Summary results of device location model performance with the machine learning approach. *The ranges of overall accuracy were computed as symmetric 95% confidence intervals from the Binomial distribution.*

Pouch (Waist)			Pocket		
Predicted / True	Non-falls	Falls	Predicted / True	Non-falls	Falls
Non-falls	1494	107	**Non-falls**	1492	109
Falls	0	1873	**Falls**	1	1872
Hand			**All Locations**		
Predicted / True	Non-falls	Falls	Predicted / True	Non-falls	Falls
Non-falls	1581	20	**Non-falls**	1582	19
Falls	117	1756	**Falls**	19	1854

Table 9. Confusion matrices for device location models with the machine learning approach.

Comparison with a Threshold-Based Algorithm

There are existing models for fall detection which do not use machine learning which we consider here for comparison. For this purpose, we built a simple threshold-based classifier which counts a clip as a fall if a minimum acceleration threshold is reached (e.g. during freefall) and a maximum acceleration threshold is also reached (e.g. during impact) - a straightforward alternative to a machine learning approach with thousands of parameters. The lower and upper thresholds were set by measuring properties of the falls in the training sample; the thresholds were the highest of all minimum acceleration values and the lowest of all maximum acceleration values. We note that higher accuracies could be possible by adapting the thresholds based on accuracies observed in a validation set, but we chose these thresholds for convenience as we expected them to give a high accuracy. We used the same validation approach as the device location validation but applied the threshold-based algorithm for comparison. The threshold-based model significantly underperformed in overall

classification accuracy when trained on data from all locations (92.72% vs 98.91%) and also underperformed when compared to the machine learning classification approach for each individual location model (Table 10 and Table 11).

Model	Precision		Recall		Overall Accuracy
	Fall	Non-Fall	Fall	Non-Fall	
Pouch (Waist)	91.31%	97.60%	98.13%	89.07%	93.96% (93.11%-94.72%)
Pocket	91.41%	90.62%	92.04%	89.88%	91.05% (90.05%-91.98%)
Hand	92.93%	99.59%	99.68%	91.13%	95.74% (95.01%-96.39%)
All Locations	88.10%	100.0%	100.0%	84.20%	92.72% (91.80%-93.56%)

Table 10. Summary results of device location models performance with threshold-based algorithm. *The ranges of overall accuracy were computed as symmetric 95% confidence intervals from the Binomial distribution.*

Pouch (Waist)		
Predicted / True	**Non-falls**	**Falls**
Non-falls	1426	175
Falls	35	1838

Pocket		
Predicted / True	**Non-falls**	**Falls**
Non-falls	1439	162
Falls	149	1724

Hand		
Predicted / True	**Non-falls**	**Falls**
Non-falls	1459	142
Falls	6	1867

All Locations		
Predicted / True	**Non-falls**	**Falls**
Non-falls	1348	253
Falls	0	1873

Table 11. Confusion matrices for device location models with the threshold-based algorithm.

Evaluation Using Real-World Data

To evaluate how the model performs in real-world scenarios we applied device location models to the available home-acquired data which represents only non-falls. The model which was trained on data from all locations performed best with only 12 false alarms per day (0.06%-0.09% false positive rate). Pocket and pouch models yielded approximately the same results with 50 and 56 false alarms per day (0.27%-0.32% and 0.30%-0.35% false positive rate) accordingly. Hand models had the highest false positive rate (0.91%-1.01%) with 166 false alarms per day. False positive rates and average number of false alarms per day for all models are shown in Table 12.

Model	False Positive Rate	Avg. # of False Alarms per Day
Pouch (Waist)	0.32% (0.30%-0.35%)	56
Pocket	0.29% (0.27%-0.32%)	50
Hand	0.96% (0.91%-1.01%)	166
All Locations	0.07% (0.06%-0.09%)	12

Table 12. Summary results of device location model performance on continuously recorded data outside the lab environment. *The ranges of overall accuracy were computed as symmetric 95% confidence intervals from the Binomial distribution.*

CHAPTER IV

DISCUSSION

Relation to Previous Work

It has previously been shown that a mobile phone can be used for reliable data collection and successful activity recognition (Albert et al., 2012; Albert et al., 2013; Dai et al., 2010; Anderson et al., 2007; Bieber et al., 2009; Brezmes et al., 2009; Gyorbiro et al., 2009; Ketabdar and Polzehl, 2009; Kwapisz et al., 2011; Mellone et al., 2012; Ying et al., 2011; Lustrek and Kaluza, 2009). In this study we use an Android smartphone as a platform for real-time fall detection. While dedicated wearable sensors have been successfully used for fall detection (Bagala et al., 2012; Shi et al., 2012; Shany et al., 2012; Zhang et al., 2011; Ying et al., 2011; Tolkiehn et al., 2011), the use of a mobile phone can provide additional advantages. Modern smartphones have various built-in sensors for motion (accelerometer and gyroscope), altitude and pressure (barometer), location (GPS), and many others for wireless data transmission. They have great processing power which permits limited analysis to be done locally. Almost everyone knows how to use a smartphone and many people already wear them daily, making adoption and compliance straightforward.

Our analysis of different machine learning classifiers for fall detection led us to implement logistic regression with the elastic net penalty for fall detection on the phone. Our results align with Albert et al. which noted that logistic regression with the lasso

penalty had the best fall detection accuracy. One of the advantages of such regularized logistic regression approaches is the ability to handle large feature sets by automatically weighting features based on their influence on the classification accuracy and automatically removing features that do not aid in prediction. Because of this ability to handle selecting and combining large feature sets, the data-driven approach is capable of creating a more accurate model that one in which the features are hand-picked by a researcher - for example, in the use of threshold-based fall detection (Igual et al., 2013).

It has been demonstrated that because subjects with mobility impairments move differently than healthy controls, accuracy of activity recognition can differ substantially for mobility-impaired populations (Albert, Toledo et al., 2012; Lonini et al., 2016). People with lower-limb amputations are especially vulnerable to falling. For this reason, we considered the possibility that amputee falls may differ from healthy control falls, and so recruited both populations under this assumption. However, the analysis has shown no significant effect on accuracy when the model was trained and tested on data from these different populations. Consequently, the data collected from healthy controls can be used to develop a fall detection model intended for amputee use.

The majority of fall detection systems that exist today use a fixed location on the body for the wearable device, most often the waist, as fixing the location makes fall detection easier (Habib et al., 2014). However, in real-life scenarios it is very impractical, or almost impossible, to always carry a phone in a fixed position in the same orientation. For this reason, we analyzed data with subjects wearing the phone at 3 different locations (pouch/waist, pocket, and hand). Additionally, we did not have a fixed orientation of the phone, which makes for a more challenging classification

problem, but is more fitting for how people wear their phones in real life. The model was trained on each single location separately, or the entire data set, and tested on data from all locations. All models had an overall classification accuracy of more than 96% with the model trained with all locations performing best (98.91%). The hand model had the lowest overall accuracy of all the models, which could be expected as ADLs with a phone held in hand tended to have higher accelerations than the ones with a phone placed in a pouch or pocket. Given our results, the model trained on all device locations gives the best accuracy, and the performance is best when wearing the phone in a pouch or pocket but still works reasonably well even in the hand.

Many existing fall detection systems use threshold-based algorithms (Habib et al., 2014; Bagala et al., 2012). We compared our approach with a simple threshold algorithm that detects a fall when the minimum and maximum acceleration reach lower and upper thresholds respectively. The overall accuracy of the threshold-based algorithm was significantly lower than our machine learning approach. The supports the notion that the machine learning approach is generally more accurate than pre-selected features for classification in practice (Igual et al., 2013). In addition, threshold-based algorithms have higher false positive rates, which would make them less effective when applied to real-world data (Bagala et al., 2012).

To observe how our model would perform in real-life scenarios, we applied it to data collected from 3 amputee subjects outside of the lab for at least 48 hours. No subject fell during that period, so all the data represented non-falls. All the false positive rates were lower than 1% with the best model indicating a fall at a rate of 0.07%, giving

only 12 false alarms per 24 hours. These results are promising and bring us one step closer to the real-time deployment of the fall detection system.

Limitations

The main limitations emerge from the chosen design of this research study. The data that was used for training was collected in a controlled laboratory setting, as the safety of our subjects was a priority. It is known that real-world falls differ from instructed falls (Klenk et. al., 2010) and algorithms developed using only lab data are not as successful when applied to real-world falls (Bagala et al., 2010). Real-world data can be obtained by asking subjects to carry a phone over extended period of time and report if they fall. However, recording all the movement data, and manually parsing the fall events would be massively time consuming and logistically challenging as falls don't occur very often even in fall-prone populations. This is part of the reason this work has the goal of automating the process of recording falls-related information.

Although we were able to achieve high overall accuracies, the false positive rate in at-home use is still impractical. A production-ready real-world detection system should produce no more than few false alarms per day. For example, if the system's fall response includes contacting emergency medical services, a high false alarm rate could be very costly. The false positive rate could be readily decreased by fixing position and orientation of the mobile phone (Bagala et al., 2012). However, requiring someone to carry their phone a specific way at all times is impractical and would most likely result in lower compliance rates among the clinical population. This is why we decided to allow variability in phone placement, increasing the chance for false positives, but being more

realistic for deployment. We expect to improve the false positive rate over time by iteratively incorporating misclassified data during the early stages of deployment.

Real-Time Fall Detection Challenges

When developing a real-time fall detection system there are certain challenges that have to be addressed. As we chose to implement the system on a mobile platform, we had to consider the limitations of embedded systems such as processing power, battery life, storage size, and internet accessibility.

Since sensor information has to be tested for a fall every 5 seconds, the evaluation process (including gathering sensor data, calculating the full set of 1211 features, applying model coefficients, and calculating the fall probability) has to take less than 5 seconds. To do this in real-time, sensors readings are accumulated by a separate thread, then the complete data bundle is passed to the fall detection model. This setup allows the evaluation of a bundled 5 second clip to take only 2 seconds - fast enough to do in real-time with time to spare.

One of the main concerns about the real-time fall detection system is battery life. We only tested battery use with the Purple Robot application running in the background and nothing more, and it was able to work successfully for more than 24 hours without recharging. However, we need to test it under more realistic conditions, including when making calls, texting, using internet, etc. This would have to be considered for different phone models that subjects may want to use.

Currently, the Purple Robot application transmits all the data to a server in real-time, assuming an internet connection is available. When there is no internet connection, the data is accumulated on the device, which is limited. If there is not

enough space available on the device and data cannot be transmitted due to absence of an internet connection, it gives a user an alert recommending to find a Wi-Fi connection as soon as possible; but after the storage limits are reached, the oldest data records would be overridden by new data. To avoid the issue of losing data regardless of wireless availability, we recommend that a user connects the device to the internet at least once per day. Of course, this is only a concern in this development version; in the deployed version of the fall detection system, only useful information will be transmitted.

The storage limit is not the only possibility for losing data. As it was described in the methods section, the data must pass the data quality test to be further processed. In most cases we observed, invalid data clips occurred due to phone hardware failures which could be fixed by rebooting the system. However, in deployment it would be difficult to observe such failures; consecutive failures remain unnoticed. This would cause not only a loss of data, but can also lead to a missed response to a fall. To avoid this situation, an additional update to the Purple Robot application can be developed, so the application checks for hardware failures and either alerts a user asking them to reboot the system, or reboots the system automatically.

We also faced language specific challenges. The initial analysis was done using MATLAB. However, when features were calculated from sensor data using Java (as would be done for the Android system) some differences in the statistical libraries were observed. Since it is important that the model would be trained, tested, and deployed on the same set of features, the Java feature set was adapted for offline analysis. Additionally, we found differences between the current version of Java from Oracle (version 8) and the early version of Java that was used on older versions of Android.

To sum up, we were able to address most of the challenges faced, resulting in a successful real-time fall detection system. While additional testing and some updates might be helpful, overall, the fall detection system is reliable.

Future Efforts

Fall detection is just a single part of the plan to deploy a system that would be able to detect a fall, classify its type, provide environmental data analysis, and contact the user or others for help if needed. Below we describe the next steps that the Rehabilitation Technologies and Outcomes lab will do to forward this plan.

One of the main goals is to obtain more amputee data of real-world daily movements and real-world falls. With that purpose we will ask a few subjects to carry a phone on a daily basis for an extended period of time (at least 6 months). During this period the data will be carefully monitored. We expect in this stage to get a few false alarms per day for each subject, so no medical intervention is planned. We will follow up with subjects once a week confirming they did not fall in that time. In this way, the real-life falls (if they occur) and fall-like data (false positives) will be collected that will help to tune the detection algorithm for later deployment.

Additionally, analysis of pre-fall and post-fall activities will be performed. Activity recognition could be applied to the 15 minute period before and after the fall, which can help to evaluate the accuracy or severity of the detected fall event (e.g. is the person not moving afterward). The data about GPS location, speed, and weather can be collected for additional analysis. We expect that this valuable information would aid in understanding the reasons and circumstances behind the real-world falls in the amputee population, which can lead to better prevention strategies in the future.

We also plan to develop classification models for the type of fall that will run on the server. Pilot efforts on fall classification have already been done (Albert et al., 2012), and these efforts can be extended for the deployed system. Knowing the type of the fall and circumstances that led to a specific type of the fall can also help in developing new fall-prevention strategies.

Last but not least, the user interface will be drastically improved, so it is easy and intuitive to use. It is likely that a separate application will be developed that would communicate with the Purple Robot background process, which has many other functions that would be confusing to the users. Developing a user-friendly interface will likely be an iterative process based on user experiences, but will be critical to achieve the best compliance for long-term use.

Conclusion

In this research study we developed a smartphone-based real-time fall detection system using machine learning techniques designed specifically for subjects with lower-limb amputations. We explored the effects of subject population (amputees vs healthy controls) and device location (pouch, pocket, hand) on the model's accuracy. The results demonstrated that both amputees and healthy controls fall in similar ways, therefore the falls data collected from healthy volunteers was also used for models applied to amputee falls. The location analysis suggests that the most robust classifier is created by using training data from different locations on the body. In addition, we compared our model to a simple threshold-based approach, demonstrating as expected that the use of machine learning for fall detection can increase accuracy. Our best detection model was 98.9% accurate on lab-acquired data and still led to false positive

rates on real-world amputee data that are not yet ready for deployment (a 0.07% false positive rate equating to 12 false alarms per day). However, this machine learning approach is more flexible design approach, with a clearer path to further improve fall detection accuracy.

With skyrocketing rates of vascular diseases like diabetes, amputations are becoming more and more prevalent, with lower-limb amputations expected to double in 40 years (Ziegler-Graham et al., 2008). Additionally, the amputee population not only has a higher risk of falls as the general population, but this risk further increases with age leading to potentially life-threatening consequences. Addressing falls in mobility-impaired populations is a critical healthcare concern. We believe that the developed fall detection system can improve real-time response to falls, but can also be a boon to fall prevention research. Data associated with falls (e.g. how they fell and conditions precipitating the fall) would help to better understand the causes of falls in the amputee population. This would not only suggest clinical guidance on fall prevention, but also potentially influence prosthesis design, ultimately improving quality of life.

REFERENCE LIST

Albert, M. V., Kording, K., Herrmann, M., & Jayaraman, A. (2012). Fall classification by machine learning using mobile phones. PLoS ONE, 7(5), 3–8

Albert, M. V., McCarthy, C., Valentin, J., Herrmann, M., Kording, K., & Jayaraman, A. (2013). Monitoring Functional Capability of Individuals with Lower Limb Amputations Using Mobile Phones. PLoS ONE, 8(6), e65340

Albert, M. V, Toledo, S., Shapiro, M., & Koerding, K. (2012). Using mobile phones for activity recognition in Parkinsonís patients. Frontiers in Neurology, 3

Anderson, I., Maitland, J., Sherwood, S., Barkhuus, L., Chalmers, M., Hall, M., … Muller, H. (2007). Shakra: Tracking and Sharing Daily Activity Levels with Unaugmented Mobile Phones. Mobile Networks and Applications, 12(2–3), 185–199

Bagala, F., Becker, C., Cappello, A., Chiari, L., Aminian, K., Hausdorff, J. M., … Klenk, J. (2012). Evaluation of Accelerometer-Based Fall Detection Algorithms on Real-World Falls. PLoS ONE, 7(5), e37062–e37062

Bieber, G., Voskamp, J., & Urban, B. (2009). Activity Recognition for Everyday Life on Mobile Phones. Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments. In C. Stephanidis (Ed.) (Vol. 5615, pp. 289–296). Springer Berlin / Heidelberg

Brezmes, T., Gorricho, J.-L., & Cotrina, J. (2009). Activity Recognition from Accelerometer Data on a Mobile Phone. In S. Omatu, M. Rocha, J. Bravo, F. Fernández, E. Corchado, A. Bustillo, & J. Corchado (Eds.), IWANN ’09 Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living (Vol. 5518, pp. 796–799). Springer Berlin / Heidelberg

Dai, J., Bai, X., Yang, Z., Shen, Z., & Xuan, D. (2010). Mobile phone-based pervasive fall detection. Personal and Ubiquitous Computing, 14(7), 633–643

Delbaere, K., Crombez, G., Vanderstraeten, G., Willems, T., & Cambier, D. (2004). Fear-related avoidance of activities, falls and physical frailty. A prospective community-based cohort study. Age and Ageing, 33(4), 368–373

Fulks, J.S., Fallon, F., King, W., Shields, G., Beaumont, N., Ward-Lonergan, J. (2002). Accidents and Falls in Later Life. Generations Rev., 12(3), 2–3

Gyorbiro, N., Fabian, A., & Homanyi, G. (2009). An Activity Recognition System For Mobile Phones. Mobile Networks and Applications, 14(1), 82–91

Habib, M., Mohktar, M., Kamaruzzaman, S., Lim, K., Pin, T., & Ibrahim, F. (2014). Smartphone-Based Solutions for Fall Detection and Prevention: Challenges and Open Issues. Sensors, 14(4), 7181–7208

Jayaraman, A. (2014). Understanding Real-Life Falls in Amputees using Mobile Phone Technology. Project # 1R01EB019406-01A1 proposal. Retrieved February 19, 2017, from https://projectreporter.nih.gov/project_info_description.cfm?aid=8738041&icde=21980274

Ketabdar, H., & Polzehl, T. (2009). Fall and emergency detection with mobile phones. Proceedings of the 11th International ACM SIGACCESS Conference on Computers and Accessibility. Pittsburgh, Pennsylvania, USA: ACM

Klenk, J., Becker, C., Lieken, F., Nicolai, S., Maetzler, W., Alt, W., … Lindemann, U. (2010). Medical Engineering & Physics Comparison of acceleration signals of simulated and real-world backward falls. Medical Engineering and Physics, 33(3), 368–373

Kramarow, E., Chen, L., Hedegaard, H., Warner, M. (2015). Deaths from Unintentional Injury Among Adults Aged 65 and Over: United States, 2000–2013. National Center for Health Statistics. Retrieved April 13, 2017, from https://www.cdc.gov/nchs/data/databriefs/db199.htm

Kulkarni, J. R., Collin, C., & Collin, J. (1996). Mobility after lower-limb amputation. British Journal of Surgery, 83(1), 134–134

Kwapisz, J. R., Weiss, G. M., & Moore, S. A. (2011). Activity recognition using cell phone accelerometers. SIGKDD Explor. Newsl., 12(2), 74–82

Limb Loss Statistics | Amputee Coalition. (n.d.). Retrieved January 16, 2017, from http://www.amputee-coalition.org/limb-loss-resource-center/resources-by-topic/limb-loss-statistics/limb-loss-statistics

Lonini, L., Gupta, A., Kording, K., & Jayaraman, A. (2016). Activity recognition in patients with lower limb impairments: Do we need training data from each patient? In 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) (pp. 3265–3268)

Lustrek, M., & Kaluza, B. (2009). Fall Detection and Activity Recognition with Machine Learning. Informatica, 33(3), 205–212

MacBride, A., Rogers, J., Whylie, B., & Freeman, S. J. (1980). Psychosocial factors in the rehabilitation of elderly amputees. Psychosomatics, 21(3), 258–61, 265

Mann, R., Birks, Y., Hall, J., Torgerson, D., & Watt, I. (2006). Exploring the relationship between fear of falling and neuroticism: a cross-sectional study in community-dwelling women over 70. Age and Ageing, 35(2), 143–147

Mellone, S., Tacconi, C., Schwickert, L., Klenk, J., Becker, C., & Chiari, L. (2012). Smartphone-based solutions for fall detection and prevention: the FARSEEING approach. Zeitschrift F{ü}r Gerontologie Und Geriatrie, 45(8), 722–727

Miller, W. C., Speechley, M., & Deathe, B. (n.d.). The Prevalence and Risk Factors of Falling and Fear of Falling Among Lower Extremity Amputees

Purple Robot - CBITs TECH Web Site. (n.d.). Retrieved February 19, 2017, from https://tech.cbits.northwestern.edu/purple-robot

Ryynanen, O. P., Kivela, S. L., Honkanen, R., & Laippala, P. (1992). Falls and Lying Helpless in the Elderly. Z Gerontology, 25(4), 278–282

Spice, C. L., Morotti, W., George, S., Dent, T. H. S., Rose, J., Harris, S., & Gordon, C. J. (2009). The Winchester falls project: a randomised controlled trial of secondary prevention of falls in older people. Age and Ageing, 38(1), 33–40

Sixsmith, A., and Johnson, N. (2004). A smart sensor to detect the falls of the elderly. IEEE Pervasive Computing, 3(2), 42–47

Shany, T., Redmond, S. J., Narayanan, M. R., & Lovell, N. H. (2012). Sensors-Based Wearable Systems for Monitoring of Human Movement and Falls. IEEE Sensors Journal, 12(3), 658–670

Shi, Y., Shi, Y., & Wang, X. (2012). Fall Detection on Mobile Phones Using Features from a Five-Phase Model. In 2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing (pp. 951–956)

Tolkiehn, M., Atallah, L., Lo, B., & Yang, G.-Z. (2011). Direction sensitive fall detection using a triaxial accelerometer and a barometric pressure sensor. In Engineering in Medicine and Biology Society,EMBC, 2011 Annual International Conference of the IEEE (pp. 369–372)

Vellas, B. J., Wayne, S. J., Romero, L. J., Baumgartner, R. N., & Garry, P. J. (1997). Fear of falling and restriction of mobility in elderly fallers. Age and Ageing, 26(3), 189–193

Ying, L., Redmond, S. J., Narayanan, M. R., & Lovell, N. H. (2011). Classification between non-multiple fallers and multiple fallers using a triaxial accelerometry-based system. In Engineering in Medicine and Biology Society,EMBC, 2011 Annual International Conference of the IEEE (pp. 1499–1502)

Ziegler-Graham, K., MacKenzie, E. J., Ephraim, P. L., Travison, T. G., & Brookmeyer, R. (2008). Estimating the Prevalence of Limb Loss in the United States: 2005 to 2050. Archives of Physical Medicine and Rehabilitation, 89(3), 422–429. https://doi.org/10.1016/j.apmr.2007.11.005

Zhang Z., Kapoor, U., Narayanan, M., Lovell, N. H., & Redmond, S. J. (2011). Design of an unobtrusive wireless sensor network for nighttime falls detection. In 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (pp. 5275–5278)

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic-net. Journal of the Royal Statistical Society, 67(2), 301–320

VITA

Ilona Shparii originally comes from the city of Chervonohrad which is located in the Lviv region of Ukraine. She earned her Bachelor of Science degree in Economics from Ivan Franko National University of Lviv with Honors in 2010. A year later she defended her Master of Science thesis in Accounting and Audit and also received Diploma with Honors from Ivan Franko National University of Lviv.

Ilona came to the United States of America in 2012 where she received training related to business, leadership, and technology. At that time she discovered her true passion for creating solutions by using technology and has been developing skills in this field ever since.

In 2015 Ilona joined the Computer Science Master's program at Loyola University Chicago. She was first introduced to machine learning as she took a course with the same title. As she received a joint Loyola University Chicago and Rehabilitation Institute of Chicago (RIC) $12,000 fellowship she got an opportunity to apply her machine learning skills to the project she was working on. She received a Poster Award Finalist at the end of 2016 RIC summer internship. These research efforts finally resulted in this Master's thesis. At her graduation, she was awarded the Computer Science Department's highest graduate student honor, the Dijkstra award, given to only one student each year.